

FIG. 3

2/9

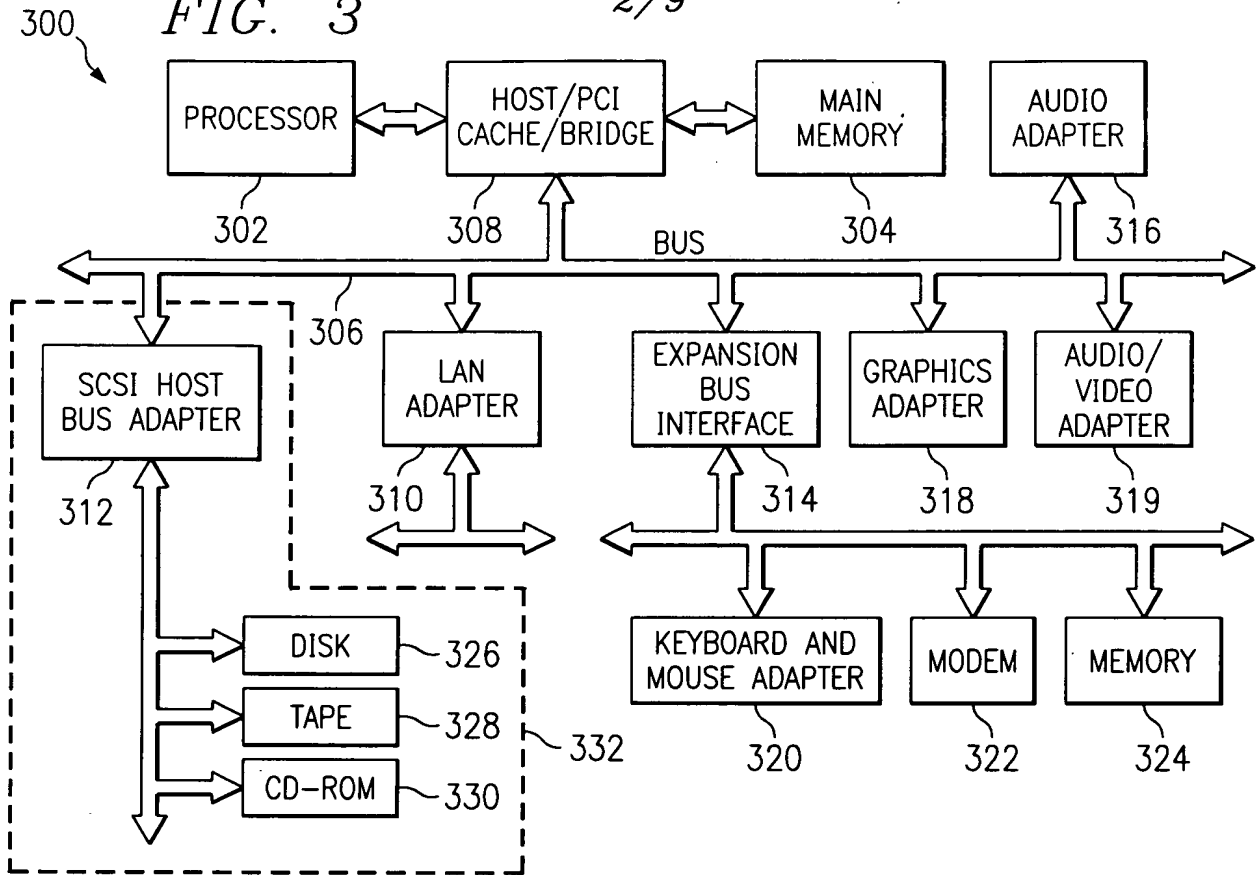
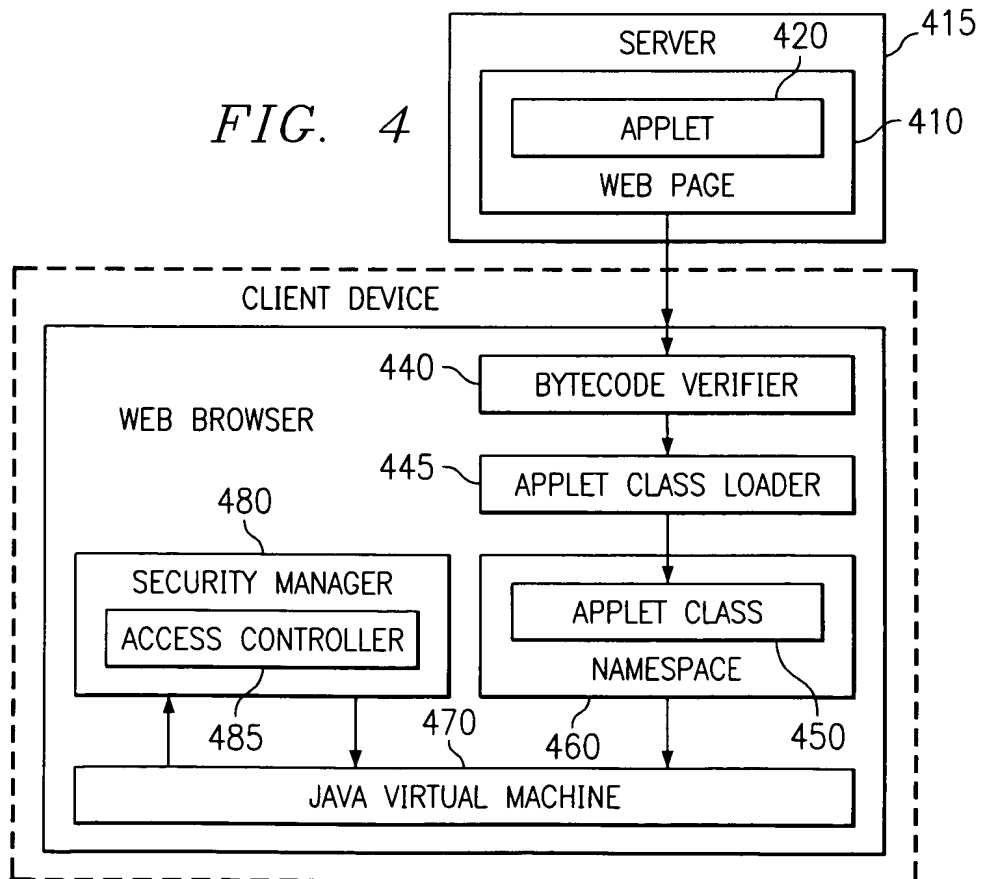


FIG. 4



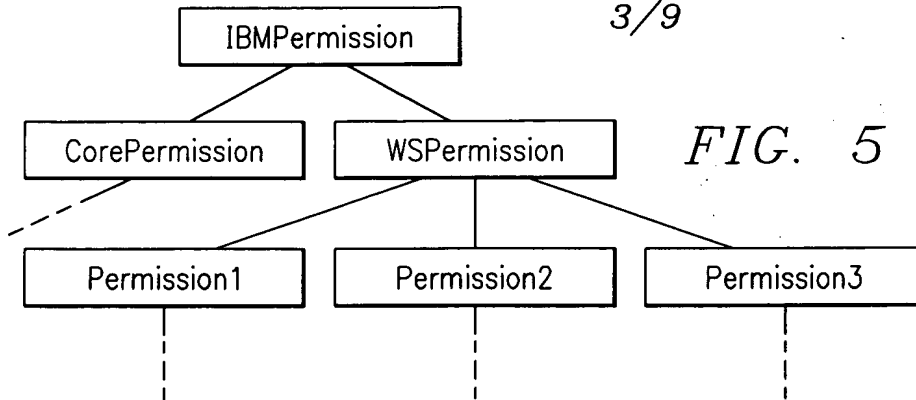


FIG. 5

FIG. 6

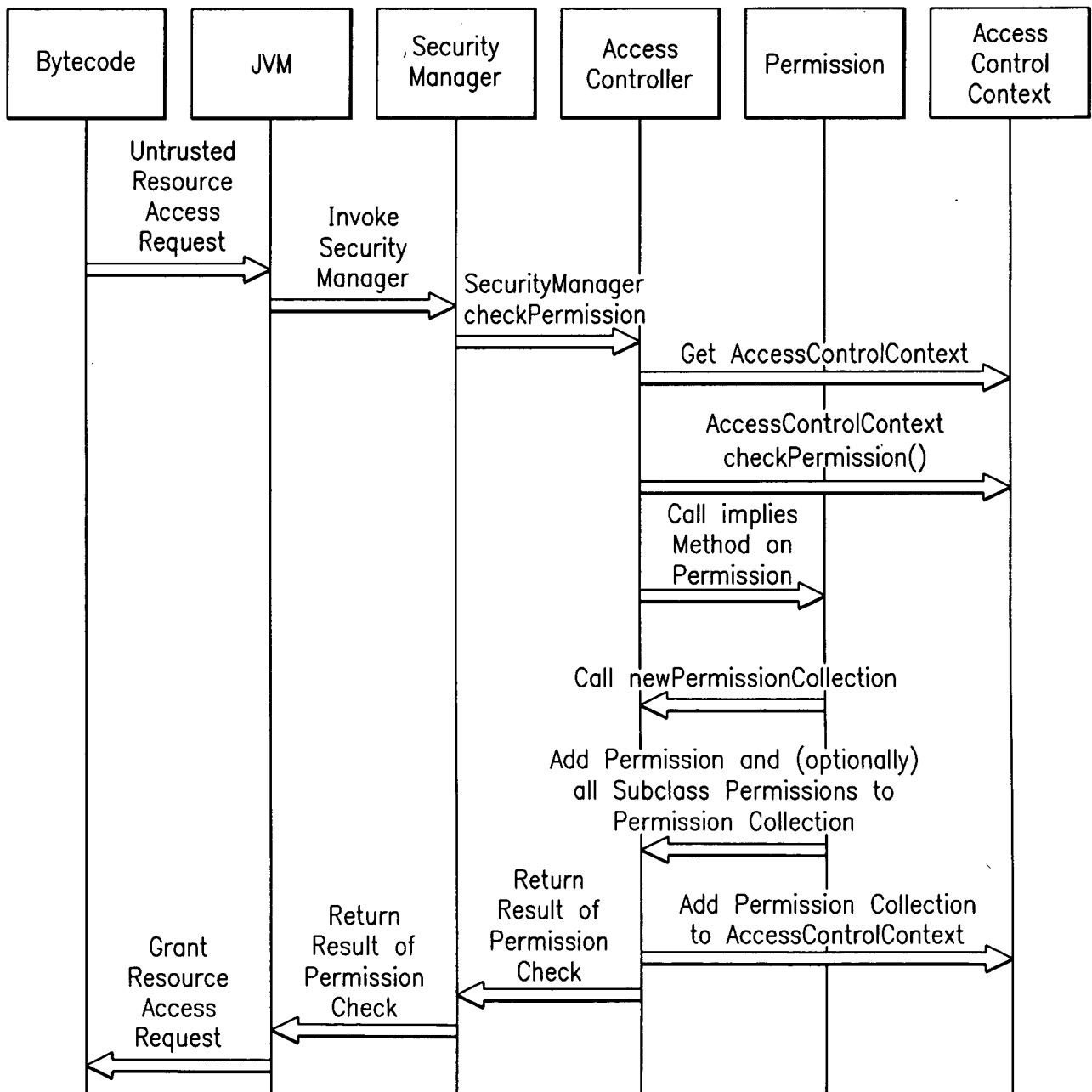


FIG. 7A

```
import java.security.BasicPermission;
import java.security.Permission;
import java.security.PermissionCollection;
import java.util.Hashtable;
import java.util.Enumeration;

public class IBMPermission extends BasicPermission
{
    public IBMPermission()
    {
        super (" ");
        System.out.println("Constructor IBMPermission() called");
    }
    public IBMPermission(String target)
    {
        super(target);
        System.out.println("Constructor IBMPermission(target) called");
    }
    public IBMPermission(String target, String actions)
    {
        super(target, actions);
        System.out.println("Constructor IBMPermission(target, actions) called");
    }
    public boolean implies(Permission perm)
    {
        System.out.println("IBMPermission.implies() called");
        if (perm instanceof IBMPermission)
            return true;
        return false;
    }
    public PermissionCollection newPermissionCollection()
    {
        return new IBMPermissionCollection();
    }
}
```

```

final class IBMPermissionCollection extends PermissionCollection
    implements java.io.Serializable
{
    private Hashtable permissions;

    public IBMPermissionCollection()
    {
        permissions = new Hashtable();
    }

    public void add(Permission permission)
    {
        if (! (permission instanceof IBMPermission))
            throw new IllegalArgumentException("Invalid Permission: " +
                                             permission);

        IBMPermission ibmp = (IBMPermission) permission;
        permissions.put(ibmp.getName(), permission);
    }

    public boolean implies (Permission permission)
    {
        if (! (permission instanceof IBMPermission))
            return false;

        System.out.println("permission instanceof IBMPermission == true");

        IBMPermission ibmp = (IBMPermission) permission;
        String permName = ibmp.getName();
        Permission x = (Permission) permissions.get(permName);

        if (x != null)
        {
            System.out.println("We have a direct hit! " + x.getName());
            return x.implies(permission);
        }

        Enumeration permEnum = permissions.elements();
        while (permEnum.hasMoreElements())
        {
            x = (IBMPermission) permEnum.nextElement();
            System.out.println(x.getName());

            if (x.implies(permission))
                return true;
        }

        return false;
    }

    public enumeration elements()
    {
        return permissions.elements();
    }
}

```

FIG. 7B

Method and Apparatus for Implementing Permission Based
Access Control Through Permission Type Inheritance

import java.security.PermissionCollection; 6/9
import java.security.AccessController;
import java.security.AccessControlContext;
import java.security.AccessControlException;

FIG. 7C

```

public class WSPermission extends IBMPermission
{
    public WSPermission(String target)
    {
        super(target);
        System.out.println("Constructor WSPermission(target) called");
    }

    public WSPermission(String target, String actions)
    {
        super(target, actions);
        System.out.println("Constructor WSPermission(target, actions) called");
    }

    public WSPermission()
    {
        super("");
        System.out.println("Constructor WSPermission() called");
    }

    /**
     * Returns a new IBMPermissionCollection object for storing IBMPermission
     * objects.
     * <p>
     * An IBMPermissionCollection stores a collection of
     * IBMPermission permissions.
     * <p>
     * IBMPermission objects must be stored in a manner that allows them
     * to be inserted in any order, but that also enables the
     * PermissionCollection <code>implies</code> method
     * to be implemented in an efficient (and consistent) manner.
     *
     * @return a new IBMPermissionCollection object suitable for
     *         storing IBMPermission's.
     */
    public PermissionCollection newPermissionCollection()
    {
        System.out.println("newPermissionCollection() was called");
        IBMPermissionCollection ibmPC = new IBMPermissionCollection();
        // the code here checks if an IBMPermissionCollection has been granted.
        // If yes, then the PermissionCollection returned by this
        // method should contain a WSPermission.
        AccessControlContext acc = AccessController.getContext();
        try
        {
            acc.checkPermission(new IBMPermission("PermissionTest"));
            ibmPC.add(new WSPermission("PermissionTest"));
        }
        catch (AccessControlException ace)
        {
            System.out.println("IBMPermission WAS NOT GRANTED");
        }
        return ibmPC;
    }
}

```

FIG. 8

```
import java.io.*;

public class PermissionTest
{
    public static void main(String args[])
    {
        try
        {
            SecurityManager sm = System.getSecurityManager();

            if (sm != null)
            {
                System.out.println("SecurityManager is checking for " +
                                   "WSPermission");

                sm.checkPermission(new WSPermission("PermissionTest"));
            }

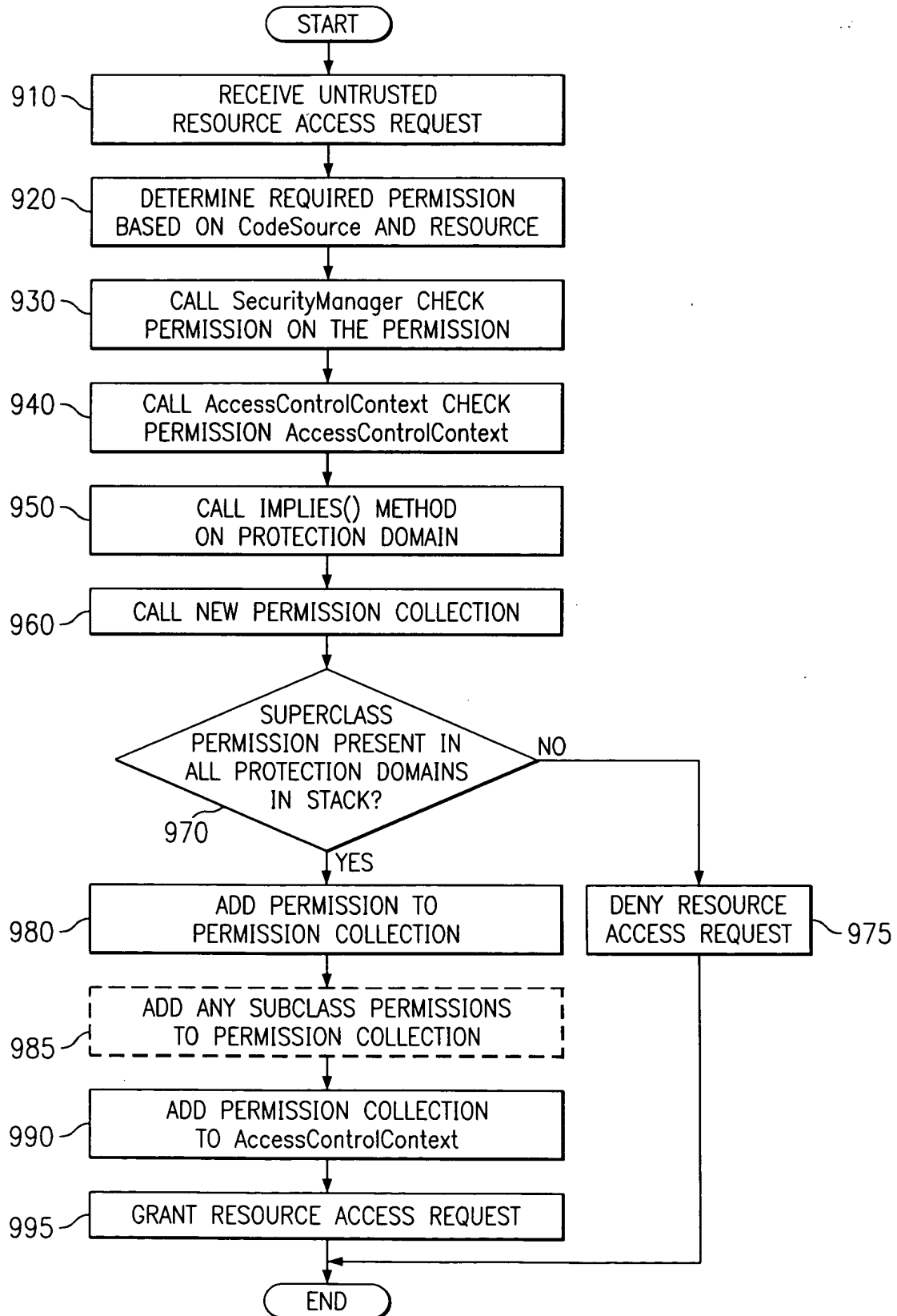
            System.out.println("WSPermission was granted. " +
                               "Permission testing
worked.\n\n\n");

            File inputFile = new File("C:\\winzip.log");
            FileInputStream fis = new FileInputStream(inputFile);
            InputStreamReader isr = new InputStreamReader(fis);
            BufferedReader br = new BufferedReader(isr);

            String lineRead;
            while ((lineRead = br.readLine()) != null)
                System.out.println(lineRead);
        }

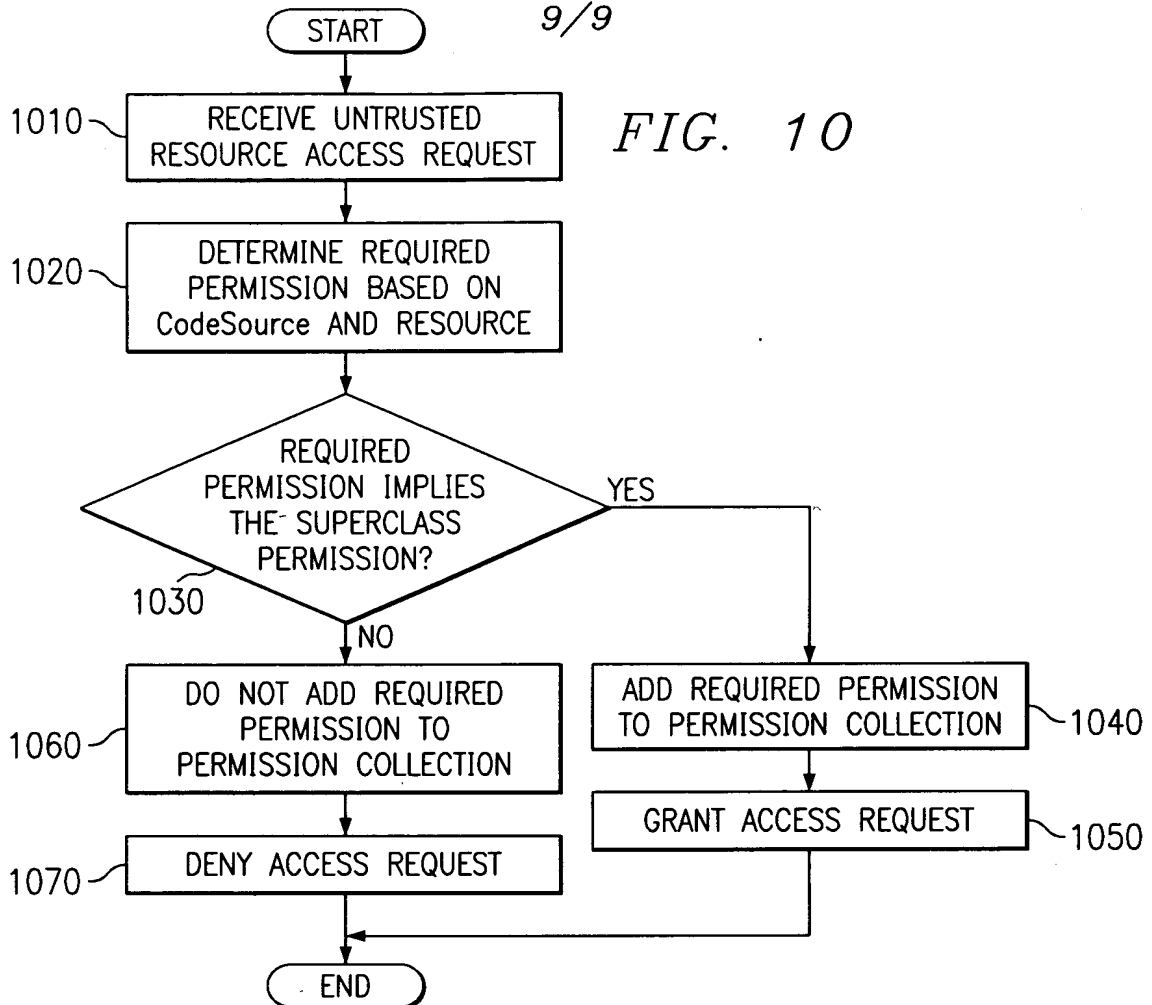
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

FIG. 9



Method and Apparatus for Implementing Permission Based
Access Control Through Permission Type Inheritance

9/9

*FIG. 11*

```

package sun.security.provider;

import java.security.PermissionCollection;
import java.security.CodeSource;
import IBMPermission;
import WSPermission;

public class MarcoPolicy extends PolicyFile
{
    public PermissionCollection getPermissions(CodeSource codesource)
    {
        PermissionCollection pc = super.getPermissions(codesource);
        if (pc == null)
            return null;

        if (pc.implies(new IBMPermission("PermissionTest")))
            pc.add(new WSPermission("PermissionTest"));

        return pc;
    }
}
  
```